Cours développement mobile en Swift

Construire une TableView

Voici les étapes pour construire une TableView moderne et paramètrable. On fait l'hypothèse que l'on travaille avec un ViewController dont le nom est `MainViewController`.

- Dans Interface Builder : placer une tableView dans la vue. Elle n'a pas besoin d'occuper toute la vue.
- Toujours dans IB, ajouter un objet de type TableViewCell dans la tableView
- Dans IB : La hauteur de la cellule peut être modifiée avec le paramètre `row height` dans le size inspector de la tableView
- Ajouter les objets dans la cellule, le plus souvent des UILabel et des UIImage. Les placer correctement via l'auto layout.
- Dans le fichier du viewController, ajouter un IBOutlet, par exemple `myTableView` de type `UITableView`
- Relier la tableView avec cet IBOutlet dans Interface Builder.
- Dans IB : ajouter l'identifier "CustomCell" à la Table View Cell
- Dans IB : relier dataSource et delegate au ViewController
- Dans un fichier CustomCell.swift créer la classe CustomCell et y placer les IBOutlets des objets inclus dans la cellule de la table. Par exemple :

```
import UIKit

class CustomCell: UITableViewCell {
    @IBOutlet  var titleLabel: UILabel!
    @IBOutlet  var myImage : UIImageView!
}
```

- Dans IB: Appliquer la classe "CustomCell" à la Cellule
- Dans IB : Avec un clic droit sur `CustomCell` dans le panneau de gauche : relier les différents outlets créés dans `CustomCell.swift` avec les objets correspondants.
- Dans le viewController, ajouter le code suivant en dehors de la définition de classe du ViewController en supposant qu'il se nomme `MainViewController`. Ici, on utilise une extension pour pouvoir éviter d'alourdir le code du MainViewController :

```
// MARK: - TableView

extension MainViewController : UITableViewDelegate, UITableViewDataSource {
    func numberOfSectionsInTableView(tableView: UITableView) -> Int {
        // Return the number of sections.
        return 1
    }

    func tableView(tableView: UITableView, numberOfRowsInSection section: Int) ->
Int {
        // A modifier, retourner le nombre de ligne dans la section
        return 0
    }

    func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSInde
xPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCellWithIdentifier("CustomCell", forIn
dexPath: indexPath) as! CustomCell
        // Ajouter la logique d'affichage du texte dans la cellule de la TableView
        // la variable indexpath.row indique la ligne selectionnée
        // on accède aux IBOutlet de la cellule avec par exemple : cell.name =

        return cell
    }

    func tableView(tableView: UITableView, didSelectRowAtIndexPath indexPath: NSIn
dexPath) {

        let selectedRow = indexPath.row
        //faire quelque chose avec selectedRow

        tableView.deselectRowAtIndexPath(indexPath, animated: false)
    }
}
```

- Compléter les fonctions précédentes pour déterminer quoi afficher dans la TableView et quoi faire quand l'utilisateur clique sur une ligne. Il faudra sans doute créer un tableau dans le `MainViewController` qui contiendra les données. Une fois ce tableau chargé, on pourra faire `myTableView.reloadData()`